Karlsruher Institut für Technologie

Programmieren Tutorium 3 – Konventionen, Kontrollstrukturen

Institut für Zeritfizierbare und Vertraünswürdige Informatiksysteme (ZVI)

Inhalt

Karlsruher Institut für Technologie

Korrektur des Übungsblattes

Konventionen
- Namen
- Whitespace

Kontrollstrukturen

Aufgaben

Korrektur des Übungsblattes

Siehe Code.

Konventionen

- Helfen, code verständlich zu halten
- Helfen programmen, die Code analysieren

Konventionen

- Helfen, code verständlich zu halten
- Helfen programmen, die Code analysieren
- Sorgen dafür, dass andere (auch ihr selbst, in 8 wochen!) nachvollziehen können, was der code tut.

Konventionen

Karlsruher Institut für Technologie

- Helfen, code verständlich zu halten
- Helfen programmen, die Code analysieren
- Sorgen dafür, dass andere (auch ihr selbst, in 8 wochen!) nachvollziehen können, was der code tut.
- *Nichteinhaltung führt zu Punktabzug!*

Namenskonventionen

- Namen in CamelCase
- Klassennamen gross (`class MyClass {...}`)
- Variablennamen klein, Namen i. d. R. Substantive
  (`int importantIntVariable;`)
- Laufvariablen mit einem buchstaben, Namen i. d. R anfangsbuchstabe des typen oder darauf folgende buchstaben
  (`for(int i = 0; i < 10; i++) {...}`)
- Methodennamen klein, Namen i. d. R. Verben
  (`int doSomething(int foo) {...}`)
- Methoden zum Lesen oder Schreiben von Attributen heißen getAttr bzw. setAttr
- Konstanten in großbuchstaben, wörter durch _ getrennt
  (`public static final int IMPORTANT_CONSTANT = 1;`)

Dateien und Inhalte

- Jede Klasse in eine eigene Datei
- Dateiname: Klassenname.java

Whitespace-Konventionen

Siehe Dokumentation zu Checkstyle 1.

Kontrollstrukturen – if

```
if (boolean) {
    // code to run if boolean is true
} else {
    // code to run otherwise
}
```

Kontrollstrukturen – Ternärer Operator

```
String s = cond ? "true" : "false";
```

$\rightarrow$ Nur für kurze Bedingungen und Zweige geeignet!

Kontrollstrukturen – switch

```
switch (var) {
    case 1:
        // code if var is 1
        break;
    case 2:
    case 3:
        // code if var is 2 or 3
        break;
    default:
        // code to run if none of the above conditions hold
        break;
}
```

Kontrollstrukturen – switch

```
switch (var) {
    case 1:
        // code if var is 1
        break;
    case 2:
    case 3:
        // code if var is 2 or 3
        break;
    default:
        // code to run if none of the above conditions hold
        break;
}
```

→ Nur für primitive Datentypen!

Kontrollstrukturen – while

```
while (condition) {
    // code to run while condition holds
}
```

→ Die Bedingung wird vor dem Schleifendurchlauf geprüft.

Kontrollstrukturen – do-while

```
do {
    // code to run while condition holds
} while (condition);
```

→ Die Bedingung wird nach dem Schleifendurchlauf geprüft.

Kontrollstrukturen – for

```
for (start; stop; step) {
    //code to run while stop condition is not false
}
```

$\rightarrow$ Zählschleife

Kontrollstrukturen – Schleifenkontrolle

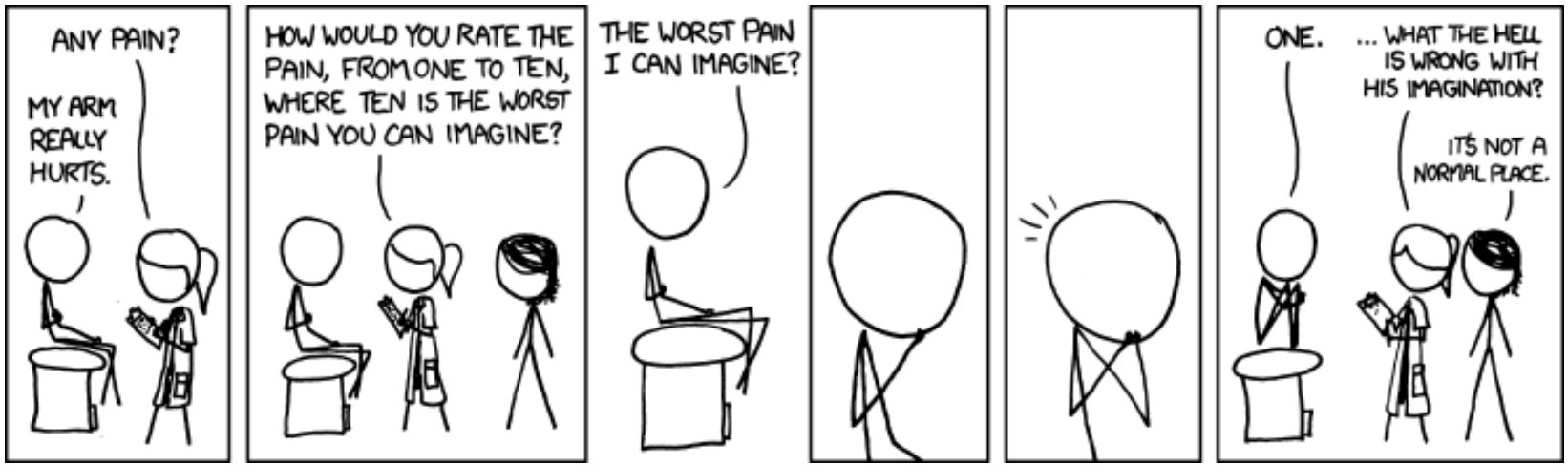
Karlsruher Institut für Technologie

Nur in ausnahmefällen, da verständlichkeit schnell leidet
Schleifenabbruch mit `break;`

Aufgabe

$$fak(0) = 1; fak(n) = n \cdot fak(n-1)$$

Ende


ANY PAIN?
MY ARM REALLY HURTS.
HOW WOULD YOU RATE THE PAIN, FROM ONE TO TEN, WHERE TEN IS THE WORST PAIN YOU CAN IMAGINE?
THE WORST PAIN I CAN IMAGINE?
ONE.
... WHAT THE HELL IS WRONG WITH HIS IMAGINATION?
IT'S NOT A NORMAL PLACE.